High-Throughput Crowdsourcing Mechanisms for Complex Tasks

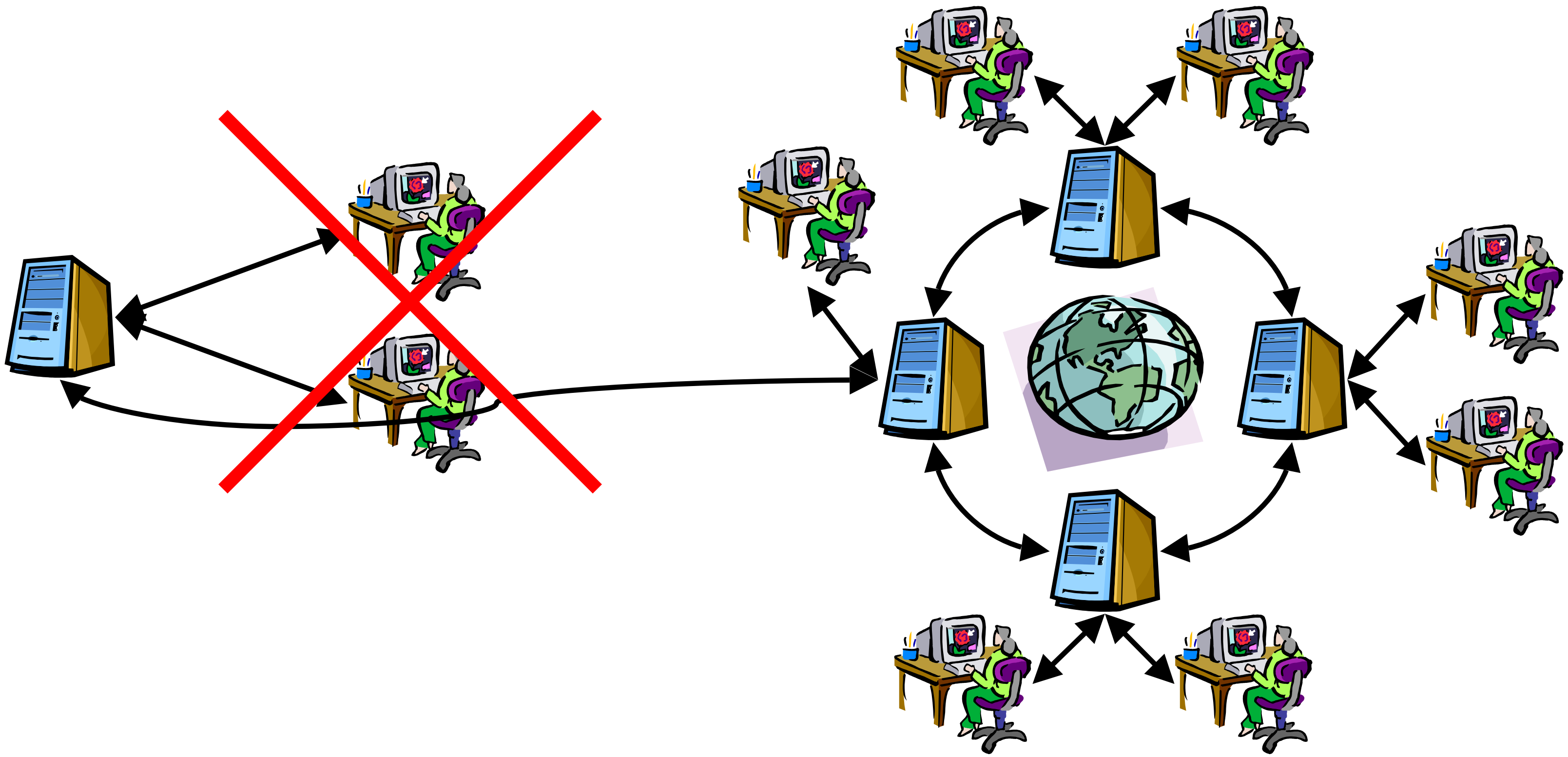
Guido Sautter, Klemens Böhm

Karlsruhe Institute of Technology

ViBRANT
Virtual Biodiversity

Crowdsourcing – What is this?

- Means to distribute work requiring human input ...
- ... to a large user community, often via the Internet



Crowdsourcing - Examples

- Real-world examples:
  - ReCAPTCHA: Double-keying words from images
  - Distributed Proofreaders: Proofreading OCR results
  - Search for Ken Fosset: Satellite image processing
  - …
- Scientific studies:
  - OntoGame: Ontology construction
  - GalaxyZoo: Classification of galaxies
  - Word sense disambiguation & other NLP tasks
  - Image labeling & classification
  - …

Crowdsourcing - Discussion

- Advantages
  - Less expensive than hiring full-time personnel
  - Faster data processing due to large workforce

- Challenges:
  - Contributing users not trained ...
  - ... hard to supervise ...
  - ... and possible dishonest

➔ Requires specific means to ensure result quality

Overview

- Crowdsourcing
- **Formal Model**
- State of the Art
- Increasing Throughput
- Evaluation
- Summary & Outlook

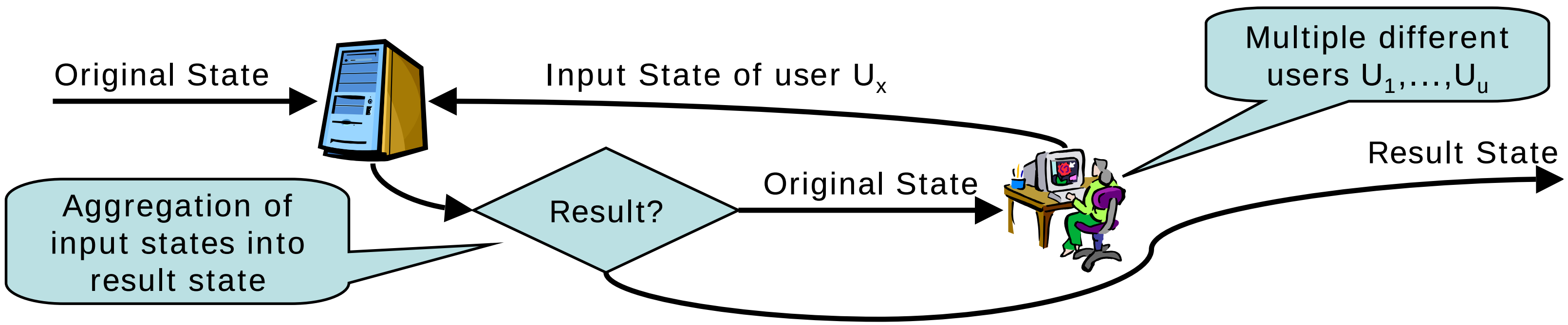
Decisions and Tasks

- Decision D: single parameter to obtain
  - Choosing appropriate option O from available options Opts(D)
  - Examples: Galaxy class, image label, transcript of word image
  - ➔ Variation in complexity

- Task T = (D_1, ..., D_d): list of decisions
  - Unit of work assigned to users
  - Decisions can be independent or connected
  - Examples: Structuring page image into blocks

States of Decisions & Tasks

- State of decision D: Option selected for D at some point
  - Original state: Option selected for D when entering system (may be pre-selected by AI algorithm or empty)
  - Input state of user U: Option user U selected for D
  - Result state: Actual state for D when leaving system
  - Correct state: Correct result for D when leaving system (what experts would agree on)
- Correspondingly for task T = (D_1, …, D_d):
  - List of respective state of decisions D_1, …, D_d



Errors

- Two types of errors:
  - **Miss Errors**: User makes or fails to correct an error
    (can always happen)
  - **Add Errors**: User falsifies correct decision
    (can only happen with algorithmic pre-decisions)
- Sources of errors:
  - **Mistakes**: Resulting from sloppiness or misjudgment
    (both miss and add errors)
  - **Cheating**: User does not bother to check thoughtfully, accepting
    everything as correct (generally miss errors)
  - Others (e.g. destructive malevolence): User falsifies on purpose
    (more theoretical, same properties as mistakes)

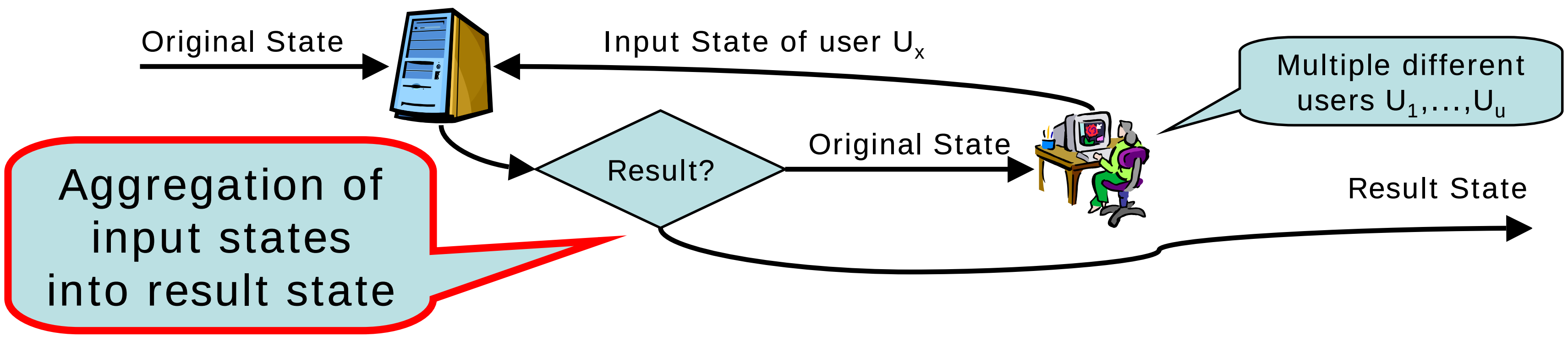
Overview

- Crowdsourcing
- Formal Model
- **State of the Art**
- Increasing Throughput
- Evaluation
- Summary & Outlook

SotA: r-Redundancy

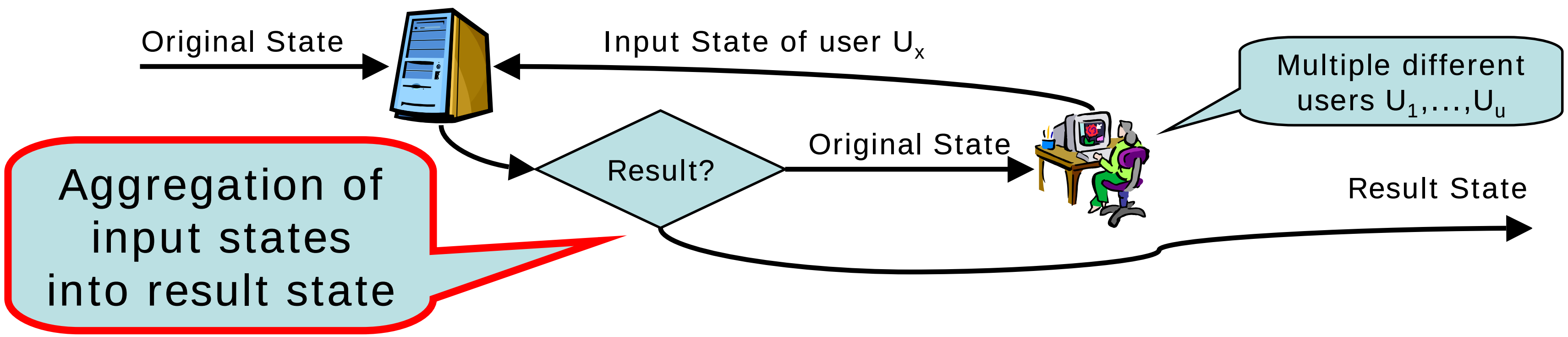
- Principle:
  - Gather input from r users
  - Result: Most frequently selected option
- Against all sorts of errors
- Value of r mostly ballpark figure so far, around 5 to 15

- Problem: sub-optimal throughput of tasks
- Studies so far mostly focused on user behavior



SotA: reCAPTCHA

- Principle:
  - Test user with decision C system knows correct result for
  - Consider actual input only if input for C correct
- Mainly against cheating
- Problem: deviates working time from decisions to process
- Problem: only works with small tasks



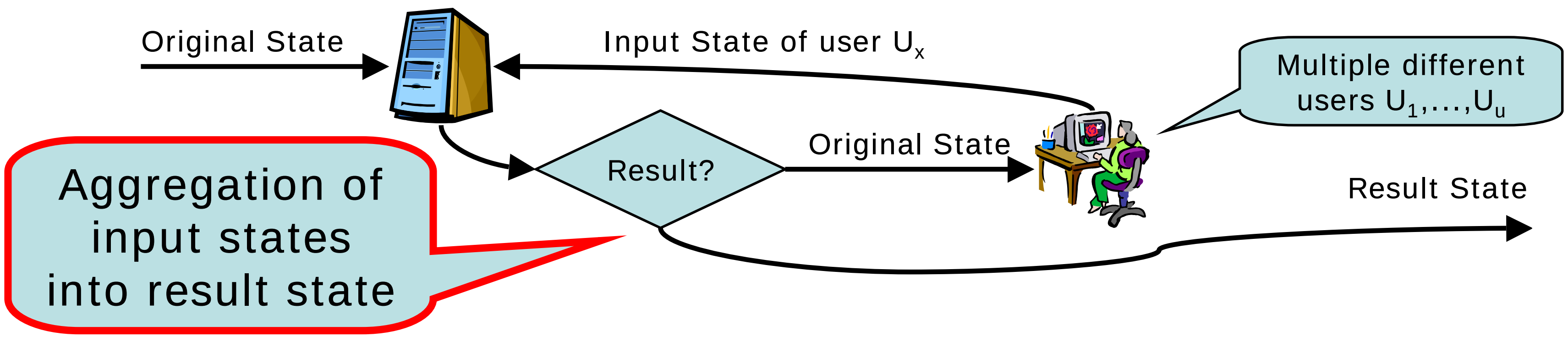
Overview

- Crowdsourcing
- Formal Model
- State of the Art
- **Increasing Throughput**
- Evaluation
- Summary & Outlook

v-Voting - Idea

- Observation: r-Redundancy gathers inputs after result is clear
- Idea for increasing throughput:
  - Stop gathering input as soon as result emerges

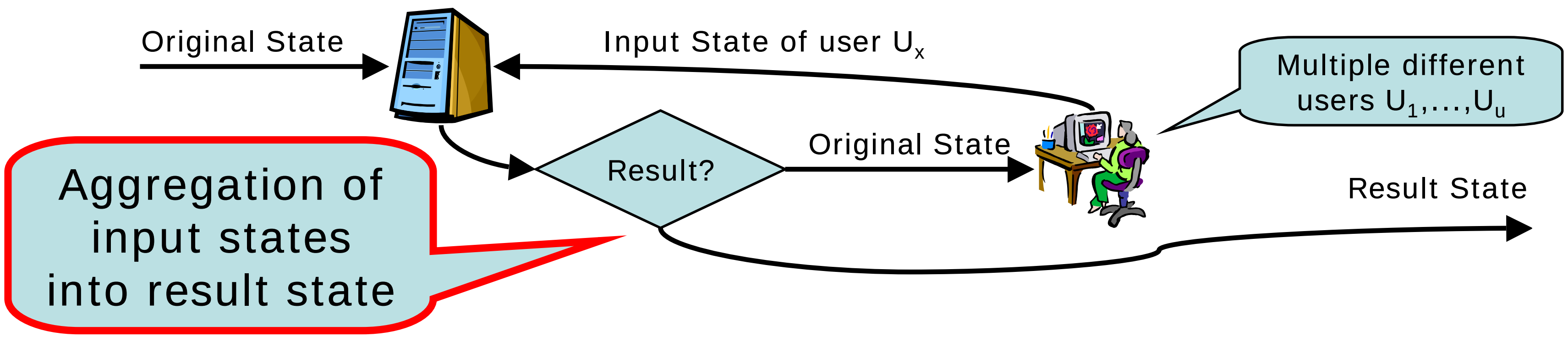
- Maintains expected result accuracy of r-Redundancy …
- … while increasing throughput



v-Voting - Mechanism

- Principle: Incremental majority vote
  - Gather input from users until v users agree
  - Result: The agreed-upon option

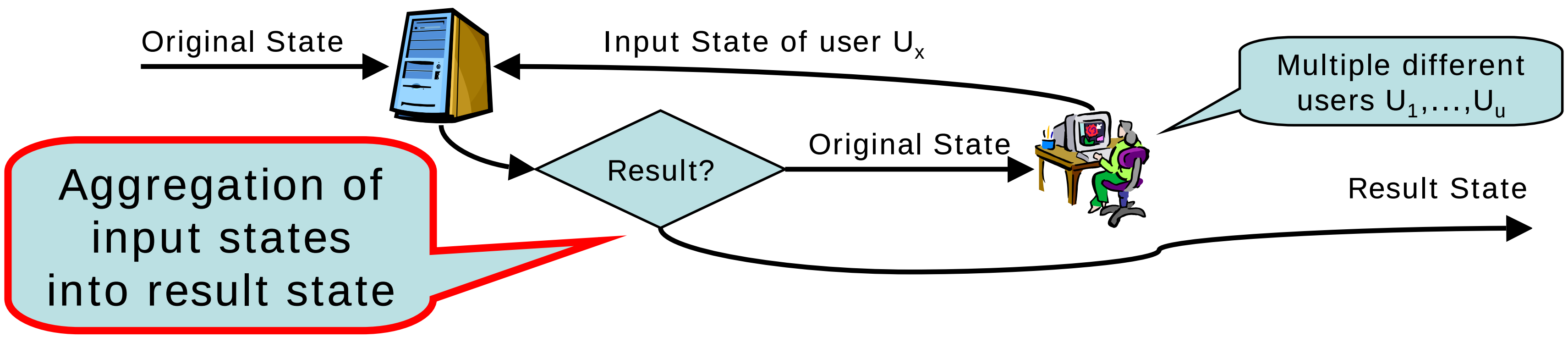
- Possible: weighting of votes depending on user ...
- ... but not investigated here



Vote Boosting - Idea

- Observation: not all users make errors with same probability
- Idea for increasing throughput:
  - Measure error rate of users in v-Voting
  - Increate weight of input from users who make few errors

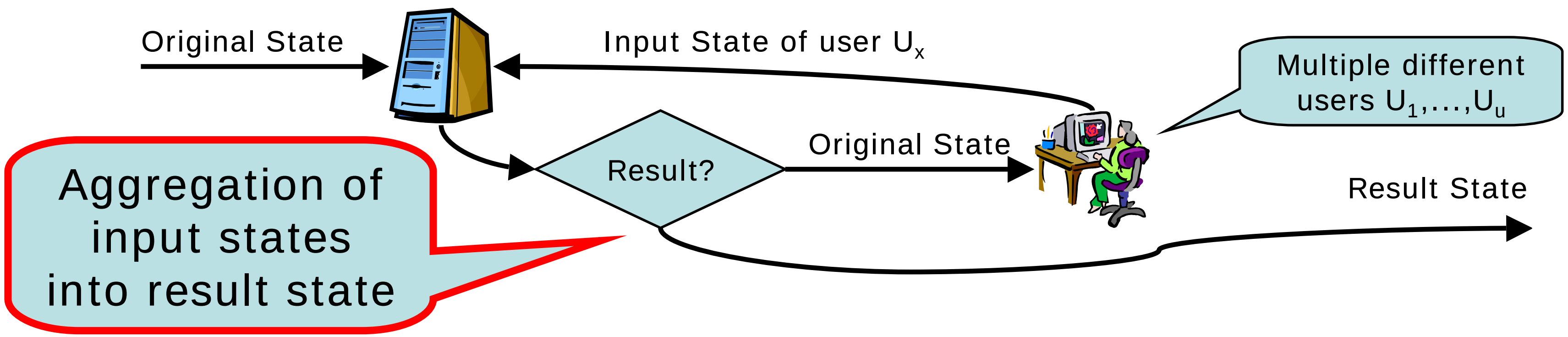
- Circumvents error prevention of v-Voting ...
- ... but increases throughput significantly



Vote Boosting - Mechanism

- Principle: If user U is first to make input on some task T
  - Take his input for correct with some *boost probability* ...
  - ... depending on how many errors U made in the past
  - Otherwise, fall back to v-Voting mechanism

- Multiple definitions of boost probability possible
- Here: based on statistical test (next slide)



Boost Probability

- Two parameters:
  - C: minimum probability of correct result
  - m: maximum probability of undeserved vote boost
- To compute / estimate: P('user U makes no error')
- Observable: number of correct inputs from user U since last error → H(U)
- *Boostability Hypothesis*: "P('user U makes no error') ≥ C"
- Approach: significance of accepting boostability hypothesis for user U based on H(U) observed correct inputs → s
- Boost probability BP(U) := m/s

Overview

- Crowdsourcing
- Formal Model
- State of the Art
- Increasing Throughput
- **Evaluation**
- Summary & Outlook

Simulation Setup

- Users: 9 populations of 1,000 each, different combinations of
  - Mean probability of making mistakes (1%, 4%, 15%)
  - Mean probability of cheating (1%, 4%, 15%)
- Tasks: 9 lists of 1,000,000 each, different combinations of
  - Options per decision (2, 3, 4)
  - Mean probability of correct initial states (80%, 90%, 95%)
- 46 methods of combining user inputs into results
  - 1-Redundancy (Base Case without any error prevention)
  - r-Redundancy with r=3,5,7
  - v-Voting with v=2,3,4 ...
  - ... each v with 14 parameter combinations for Vote Boosting

r-Redundancy vs. v-Voting

	Base Case	3-Red.	2-Voting	5-Red.	3-Voting	7-Red.	4-Voting
Remaining Error (in %)	4.25	1.11	1.01	0.48	0.46	0.27	0.27
Inputs per Task	1	3	2.36	5	3.57	7	4.75

- Aggregated over all user populations and task lists

➔ v-Voting requires fewer user inputs per task
➔ v-Voting leaves fewer errors

➔ Increase in both throughput and result accuracy

Cost of 99.5% Result Accuracy

Mean Prob. of	Mistakes		
Cheating	1 %	4 %	15 %
1 %	**1.14** (99.51%) v=2 m=8% C=92%	**1.78** (99.63%) v=2 m=4% C=96%	**3.78** (99.55%) v=3 m= 2% C=98%
4 %	**1.42** (99.57%) v=2 m=4% C=96%	**1.93** (99.51%) v=2 m=4% C=96%	**4.48** (99.51%) v=4 m=4% C=96%
15 %	**3.94** (99.65%) v=4 m=2% C=98%	**4.6** (99.61%) v=4 m=2% C=98%	**not achieved** 5.38 (98.62%) v=4 m=0

- **Inputs per task** (accuracy actually achieved)
  v controls v-Voting, m and C control Vote Boosting

➔ Cost of data quality strongly dependent on users

➔ Cheating prevention compulsory

➔ Strategy:
  - Start with conservative parameters
  - Periodically assess result quality and adjust parameters

Overview

- Crowdsourcing
- Formal Model
- State of the Art
- Increasing Throughput
- Evaluation
- **Summary & Outlook**

Summary

- Formal model of crowdsourcing systems
  - Tasks
  - Errors
  - System layout
- Better input aggregation improves results **and** throughput
  - v-Voting
  - Vote Boosting
- Cheating prevention compulsory for data quality

Outlook

- Vote Boosting: alternative definitions of boost probability
- Cheating prevention mechanisms
- Verify simulation results in real-world deployment

Guido Sautter, Klemens Böhm:

High-Throughput Crowdsourcing Mechanisms for Complex Tasks

Questions?